Lab 5

lab2boogaloo

Important restrictions

You can only use: (open, close, read, write, lseek)

No buffered library calls. Ie : fopen, fclose, fread, fwrite, fseek, fflush

All the File modes

“r” -> Opens a file for reading, the file must exist.

“w” -> Creates an empty file for writing, If a file with the same name exists, its content is erased and the file is considered as a new empty file

“a” -> Appends to a file, Writing operations will append the data at the end of the file. The file is created if it does not exist

All the File modes

“r+” -> Opens a file for reading and writing, the file must exist.

“w+” -> Creates an empty file for writing and reading, If a file with the same name exists, its content is erased and the file is considered as a new empty file

“a+” -> Appends to a file, Writing operations will append the data at the end of the file. The file is created if it does not exist. Also allows you to read, but read operations are at the start of the file.

What does fopen() do?

Following is the declaration for fopen() function.

```
FILE *fopen(const char *filename, const char *mode)
```

And the

This function returns a FILE pointer. Otherwise, NULL is returned and the global variable errno is set to indicate the error.

FILE struct

Structure :

```
typedef struct
{
 short level ;
 short token ;
 short bsize ;
 char fd ;
 unsigned flags ;
 unsigned char hold ;
 unsigned char *buffer ;
 unsigned char * curp ;
 unsigned istemp;
}FILE ;
```

Source: http://www.c4learn.com/c-programming/c-file-structure-and-file-pointer/

fread()

fread() is the buffered cousin of read(), but its buffer is actually filled by read().

4096 bytes worth of buffer in our case.

Its faster!

What does fread() do?

Declaration

Following is the declaration for fread() function.

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
```

Parameters

- **ptr** − This is the pointer to a block of memory with a minimum size of *size*nmemb* bytes.
- **size** − This is the size in bytes of each element to be read.
- **nmemb** − This is the number of elements, each one with a size of **size** bytes.
- **stream** − This is the pointer to a FILE object that specifies an input stream.

Return Value*

The total number of elements successfully read .

Source: https://www.tutorialspoint.com/c_standard_library/c_function_fread.htm

What does read() do?

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```

It reads count amount of bytes from fd into buf.

This is what you have to implement

```
size_t my_fread(void *ptr, size_t size, size_t nmemb, MY_FILE *stream) {

    return 0;
}
```

The correct implementation

You should only call “read()” whenever it's absolutely necessary.

Ie: you're not supposed to implement this lab with “read()” but rather use it to create a buffer to emulate the behaviour of “fread()”.

fwrite()

Declaration

Following is the declaration for fwrite() function.

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)
```

Parameters

- **ptr** − This is the pointer to the array of elements to be written.
- **size** − This is the size in bytes of each element to be written.
- **nmemb** − This is the number of elements, each one with a size of **size** bytes.
- **stream** − This is the pointer to a FILE object that specifies an output stream.

Return Value*

This function returns the total number of elements successfully returned

Source: https://www.tutorialspoint.com/c_standard_library/c_function_fwrite.htm

write()

```
write(int fd, const void *buf, size_t count);
```

Given fd, and the buffer *buf, write count worth of *buf into fd.

Difference in write() vs fwrite()

Write is atomic.

See: https://stackoverflow.com/questions/11414191/what-are-the-main-differences-between-fwrite-and-write

Fwrite is buffered. What does this mean?

Something something, buffer size ?

Important

If my_fread() is called

Subsequent my_fwrite() will not be called after an my_fseek() and my_fflush()

my_fflush() + my_fseek()

Figure it out yourself, Look for me if you really need help

Hint: demo